dharaneesh-2023-07-22-13-14-31

December 12, 2023

Q1.Create a DF(airlines_1987_to_2008) from this path

```
%fs ls dbfs:/databricks-datasets/asa/airlines/

(There are csv files in airlines folder. It contains 1987.csv to  2008.csv files.

Create only one DF from all the files )
```

[0]:
```
%fs ls dbfs:/databricks-datasets/asa/airlines/
```

[0]:
```
# Read all the CSV files and create the DataFrame
airlines_1987_to_2008=spark.read.option("header",True).
 ↪option("inferschema",True).csv("dbfs:/databricks-datasets/asa/airlines/*.
 ↪csv")
```

[0]:
```
# Show the DataFrame
airlines_1987_to_2008.display()
```

Q2. Create a PySpark Datatypes schema for the above DF

[0]:
```
from pyspark.sql.types import *
```

[0]:
```
# Define the schema for the DataFrame
users_schema = StructType([
    StructField("Year", IntegerType()),
    StructField("Month", IntegerType()),
    StructField("DayOfMonth", IntegerType()),
    StructField("DayOfWeek", IntegerType()),
    StructField("DepTime",  IntegerType()),
    StructField("CRSDepTime",  IntegerType()),
    StructField("ArrTime",  IntegerType()),
    StructField("CRSArrTime",  IntegerType()),
    StructField("UniqueCarrier", StringType()),
    StructField("FlightNum",  IntegerType()),
    StructField("TailNum",  IntegerType()),
    StructField("ActualElapsedTime",  IntegerType()),
    StructField("CRSElapsedTime",  IntegerType()),
    StructField("AirTime", DoubleType()),
    StructField("ArrDelay", DoubleType()),
    StructField("DepDelay", DoubleType()),
    StructField("Origin", StringType()),
    StructField("Dest", StringType()),
    StructField("Distance", DoubleType()),
    StructField("TaxiIn", DoubleType()),
    StructField("TaxiOut", DoubleType()),
    StructField("Cancelled", IntegerType()),
    StructField("CancellationCode", StringType()),
    StructField("Diverted", IntegerType()),
    StructField("CarrierDelay", DoubleType()),
    StructField("WeatherDelay", DoubleType()),
    StructField("NASDelay", DoubleType()),
    StructField("SecurityDelay", DoubleType()),
    StructField("LateAircraftDelay", DoubleType())
])
```

[0]:
```
df=spark.read.option("header",True).schema(users_schema).csv("dbfs:/
 ↪databricks-datasets/asa/airlines/*.csv")
```

Q3. View the dataframe

[0]:
```
display(df)
```

Q4. Return count of records in dataframe

[0]:
```
# Get the count of records in the DataFrame
df.count()
```

```
Out[10]: 123534969
```

Q5. Select the columns - Origin, Dest and Distance

[0]:
```
df_select=df.select("Origin", "Dest", "Distance")
```

[0]:
```
# Show the DataFrame with selected columns
df_select.display()
```

Q6. Filtering data with ‘where’ method, where Year = 2001

[0]:
```
df_filter=df.where(col("Year") == 2001)
```

[0]:
```
df_filter.display()
```

Q7. Create a new dataframe (airlines_1987_to_2008_drop_DayofMonth) exluding dropped column (“DayofMonth”)

[0]:
```
airlines_1987_to_2008_drop_DayofMonth=df.drop("DayOfMonth")
```

Q8. Display new DataFrame

[0]:
```
airlines_1987_to_2008_drop_DayofMonth.display()
```

Q9. Create column ‘Weekend’ and a new dataframe(AddNewColumn) and display

[0]:
```
from pyspark.sql.functions import *
```

[0]:
```
AddNewColumn = airlines_1987_to_2008_drop_DayofMonth.withColumn("Weekend",␣
  ↪when(col("DayOfWeek").isin(6, 7), "weekend").otherwise("no"))
```

[0]:
```
AddNewColumn.display()
```

Q10. Cast ActualElapsedTime column to integer and use printschema to verify

[0]:
```
AddNewColumn.select(col("ActualElapsedTime").cast(IntegerType()))
```

```
Out[21]: DataFrame[ActualElapsedTime: int]
```

[0]:
```
# Verify the schema using printSchema()
AddNewColumn.printSchema()
```

```
root
 |-- Year: integer (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
 |-- DepTime: integer (nullable = true)
 |-- CRSDepTime: integer (nullable = true)
 |-- ArrTime: integer (nullable = true)
 |-- CRSArrTime: integer (nullable = true)
 |-- UniqueCarrier: string (nullable = true)
 |-- FlightNum: integer (nullable = true)
 |-- TailNum: integer (nullable = true)
 |-- ActualElapsedTime: integer (nullable = true)
 |-- CRSElapsedTime: integer (nullable = true)
 |-- AirTime: double (nullable = true)
 |-- ArrDelay: double (nullable = true)
 |-- DepDelay: double (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- Distance: double (nullable = true)
 |-- TaxiIn: double (nullable = true)
 |-- TaxiOut: double (nullable = true)
 |-- Cancelled: integer (nullable = true)
 |-- CancellationCode: string (nullable = true)
 |-- Diverted: integer (nullable = true)
 |-- CarrierDelay: double (nullable = true)
 |-- WeatherDelay: double (nullable = true)
 |-- NASDelay: double (nullable = true)
 |-- SecurityDelay: double (nullable = true)
 |-- LateAircraftDelay: double (nullable = true)
 |-- Weekend: string (nullable = false)
```

Q11. Rename 'DepTime' to 'DepartureTime'

```
[0]: df2=AddNewColumn.withColumnRenamed("DepTime", "DepartureTime")
```

```
[0]: df2.display()
```

Q12.Drop duplicate rows based on Year and Month and Create new df (Drop Rows)

```
[0]: DropRows=df2.dropDuplicates(["Year","Month"])
```

```
[0]: DropRows.display()
```

Q13. Display Sort by descending order for Year Column using sort()

```
[0]: sort_df=DropRows.sort(desc("Year"))
```

```
[0]: sort_df.display()
```

Q14. Group data according to Origin and returning count

```
[0]: sort_df.groupBy("Origin").count().display()
```

Q15. Group data according to dest and finding maximum value for each 'Dest'

```
[0]: sort_df.groupBy("Dest").agg(max('Distance').alias('MaxDistance')).display()
```

Q16. Write data in Delta format

```
[0]: sort_df.write.format("delta").save("/FileStore/tables/odinschool/output/dharan")
```

```
[0]:
```